Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра програмного забезпечення комп'ютерних систем

ІНФРАСТРУКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни	
Рівень вищої освіти	***Перший (бакалаврський)***
Галузь знань	*12 Інформаційні технології*
Спеціальність	*121 Інженерія програмного забезпечення*
Освітня програма	*Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем*
Статус дисципліни	*Вибіркова*
Форма навчання	*Очна (денна)*
Рік підготовки, семестр	*4 рік підготовки, 8 семестр*
Обсяг дисципліни	*36 годин – Лекції, 18 годин – комп'ютерний практикум, 66 годин – СРС*
Семестровий контроль/ контрольні заходи	*Залік, модульна контрольна робота (МКР)*
Розклад занять	*Згідно розкладу поточного навчального року (rozklad.kpi.ua)*
Мова викладання	*Українська*
Інформація про керівника курсу / викладачів	*Лектор: к.т.н., доцент Катін Павло Юрійович,* *katin.dino@gmail.com,* *моб. +38(098)202-08-11* *Лабораторні: к.т.н., доцент Катін Павло Юрійович*
Розміщення курсу	Google classroom: https://classroom.google.com/u/0/c/MzE5ODIyMjA4MTA3?hl=ua

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчання та результати.

Вивчення дисципліни «Інфраструктура програмного забезпечення» дозволяє сформувати у студентів компетенції, необхідні для розв'язання практичних задач професійної діяльності, пов'язаної з розробленням розподіленого web-застосунку і розгортання програмної інфраструктури на базі систем контейнерної технології.

***Метою** вивчення дисципліни «Інфраструктура програмного забезпечення» є надання системних знань, вмінь і навичок про планування, розробку, тестування і будову інфраструктури програмного забезпечення на основі:*

- складної, розподіленої інфраструктури web-застосунку на базі ASP.NET Core MVC з використанням технології Docker;

- складної, розподіленої інфраструктури web-застосунку на базі Python Django з використанням технології Docker.

***Предметом** дисципліни «Інфраструктура програмного забезпечення» є процес проектування, розробки, тестування та будови розподіленої інфраструктури програмного web-застосунку з використанням системи контейнерів.*

Вивчення дисципліни «Інфраструктура програмного забезпечення» підсилює у здобувачів освіти фахових компетентностей (ФК), необхідних для розв'язання практичних задач професійної діяльності, пов'язаних з розробленням, вдосконаленням та супроводженням інтелектуальних інформаційних систем оброблення мультимедійних даних:

ФК01 Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

ФК02 Здатність брати участь у проєктуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

ФК03 Здатність розробляти архітектури, модулі та компоненти програмних систем;

ФК08 Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення;

ФК10 Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя;

ФК11 Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розроблення програмного забезпечення;

ФК12 Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення;

ФК13 Здатність обґрунтовано обирати та освоювати інструментарій з розроблення та супроводження програмного забезпечення.

Вивчення дисципліни «Інфраструктура програмного забезпечення» сприяє формуванню у студентів наступних програмних результатів навчання (ПРН) за освітньою програмою:

ПРН01 Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки;

ПРН03 Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення;

ПРН04 Знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення;

ПРН05 Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізу та математичного моделювання для розроблення програмного забезпечення;

ПРН07 Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення;

ПРН08 Знати та вміти розробляти людино-машинний інтерфейс;

ПРН09 Вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

ПРН12 Застосовувати на практиці ефективні підходи щодо проєктування програмного забезпечення;

ПРН13 Знати і застосовувати методи розроблення алгоритмів, конструювання програмного забезпечення та структур даних і знань;

ПРН15 Мотивовано обирати мови програмування та технології розробки для розв'язання завдань створення і супроводження програмного забезпечення;

ПРН16 Мати навички програмного розроблення, погодження оформлення і випуску всіх видів програмної документації;

ПРН24 Вміти проводити розрахунок економічної ефективності програмних систем.

2. Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)

Успішному вивченню дисципліни «Інфраструктура програмного забезпечення» передує вивчення дисциплін «Математичне забезпечення мультимедійних та інформаційно-пошукових систем», «Алгоритмічне забезпечення мультимедійних та інформаційно-пошукових систем», «Програмування», «Компоненти програмної інженерії» навчального плану підготовки бакалаврів за спеціальністю 121 Інженерія програмного забезпечення.

Отримані при засвоєнні дисципліни «Інфраструктура програмного забезпечення» теоретичні знання та практичні уміння забезпечують успішне виконання курсових проєктів та бакалаврських робіт за спеціальністю 121 Інженерія програмного забезпечення.

3. Зміст навчальної дисципліни

Тема 1. Технології і архітектура web-застосунків на основі типових фреймворків.

Тема 2. Програмне управління базами даних у веб-застосунках.

Тема 3. Система контейнерізації як основа створення складної розподіленої інфраструктури програмного забезпечення web-застосунку.

Тема 4. Автоматизація управління програмною інфраструктурою у системі контейнерізації.

Тема 5. Мікросервісна архітектура як основа створення програмної інфраструктури розподілених веб-застосунків.

Модульна контрольна робота.

Залік.

4. Навчальні матеріали та ресурси

Базова література:

1.Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. – Електронні текстові дані (1 файл: 2,02 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 192с. https://ela.kpi.ua/bitstream/123456789/33651/1/PIS_KL.pdf

Додаткова література:

1. Django документація [Електронний ресурс] – https://docs.djangoproject.com/en/3.2/.
2. Adam Freeman. Essential Docker for ASP.NET Core MVC.ISBN-13 (pbk): 978-1-4842-2777-0 ISBN-13 (electronic): 978-1-4842-2778-7.2017p.
3. Scott Chacon, Ben Straub. Pro Git. Версія 2.1.95-2-g8d45587, 19.01.2022.
4. Esteban Zimányi. Students: Bubacarr Jallow Shafagh Kashef. Object Relational Mapping and Entity Framework. Advanced Databases Project. 12/18/2018 p.
5. Rebeka Mukherjee. Python Scripting for System Administration.Department of Computer Science and Engineering Netaji Subhash Engineering College, Kolkata.
6. Django samples – https://docs.docker.com/samples/django/
7. Adam Freeman. Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages.2020.
8. Docker документація [Електронний ресурс] – https://docs.docker.com/get-started/.
9. Matthes E. Python Crash Course (2nd Edition) : A Hands-On, Project-Based Introduction to Programming / Eric Matthes. – San Francisco, United States: No Starch Press, US, 9. – 544 c. – (2nd Edition).
10. Thomas D. The Pragmatic Programmer : your journey to mastery, 20th Anniversary Edition / D. Thomas, A. Hunt. – Boston, United States: Pearson Education (US), 2020. – 352 c.

11. Learn Python the Hard Way : A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code – New Jersey, United States: Pearson Education (US), 2013. – 320 с.
12. Learning React : Modern Patterns for Developing React Apps – Sebastopol, United States: O'Reilly Media, Inc, USA, 2020. – 300 с.
13. Docker : Complete Guide To Docker For Beginners And Intermediates, 2020. – 140 с.
14. Docker: Up & Running : Shipping Reliable Containers in Production – Sebastopol, United States: O'Reilly Media, Inc, USA, 2018. – 347 с.
15. Docker homepage - http://www.docker.com/
16. Docker Hub - https://hub.docker.com
17. Docker blog - http://blog.docker.com/
18. Docker documentation - http://docs.docker.com/
19. Docker Getting Started Guide - http://www.docker.com/gettingstarted/
20. Docker code on GitHub - https://github.com/docker/docker
21. Docker mailing list - https://groups.google.com/forum/#!forum/docker#user
22. Docker on IRC: irc.freenode.net and channels #docker and #docker#dev
23. Docker on Twitter - http://twitter.com/docker
24. Get Docker help on Stack Overflow - http://stackoverflow.com/search?q=docker
25. Valeria Cardellini. Matteo Nardelli. Container-based virtualization: Docker. Università degli Studi di Roma "Tor Vergata" Dipartimento di Ingegneria Civile e Ingegneria Informatica Corso di Sistemi Distribuiti e Cloud Computing A.A. 2017/18.
26. Adam Freeman. Pro Angular 6 .ISBN-13 (pbk): 978-1-4842-3648-2 ISBN-13 (electronic): 978-1-4842-3649-9/2018 p.

Навчальний контент

5. Методика опанування навчальної дисципліни (освітнього компонента).

5.1 Лекційні заняття

№ з/п	*Тип навчального заняття*	*Опис навчального заняття*
Тема 1. Технології та архітектура web-застосунків на основі типових фреймворків.		
1	*Лекція 1. Технологія створення web-застосунків на основі типових фреймворків (Python Django, .NET Core).*	*Основи інфраструктури розподілених веб-застосунків. Апаратна складова інфраструктури. Системи контейнеризації для розгортання складних розподільних вебзастосунків. Технологія створення web-застосунків на основі типових фреймворків (Python Django). Технологія створення web-застосунків на основі типових фреймворків (.NET Core).* *Завдання на СРС: п. 6 №1.*
2	*Лекція 2.Програмна архітектура вебзастосунку на основі типових шаблонів (патернів) Python Django.*	*Архітектура веб-застосунку на основі патерну MVT фреймворку (Python Django). Установка Django на персональний компьютер. Установка Django контейнер Docker. Створення проєкту та налаштування серверу для роботи з Django. Створення вебзастосунку у межах проекту Django. Тестування проекту. Базові складові Django. Технології JavaSripth як елемент архітектури вебзастосунку.* *Завдання на СРС: п.6 №2.*

3	*Практикум № 1.1*	*Розробка та тестування вебзастосунку на основі типового фреймворку.* *Завдання на СРС: п.6 №3*
4	*Лекція 3. Програмна архітектура вебзастосунку на основі .NET Core.*	*Архітектура вебзастосунку на основі патерну MVC фреймворку (.NET Core). Створення вебзастосунку на основі фреймворку (.NET Core). Базові складові .NET Core. Технології JavaSripth як елемент архітектури вебзастосунку .NET Core.* *Завдання на СРС: п.6 №4.*
5	*Лекція 4. Розгортання вебзастосунку у мережі як складної програмної інфраструктури.*	*Перелік та технології розгортання вебзастосунків у мережі. Практика розгортання вебзастосунку. Використання контейнера Docker для розгортання вебзастосунку. Основи автоматизації та масштабування складної програмної інфраструктури.* *Завдання на СРС: п.6 №5.*
6	*Практикум № 1.2*	*Розробка та тестування веб-застосунку на основі типового фреймворку.* *Завдання на СРС: п.6 №6*
Тема 2. Програмне управління базами даних у веб-застосунках.		
7	*Лекція 5. Програмна архітектура типової об'єктно-реляційній проекції (Object-relational mapping(ORM)) для роботи з базою даних типових розподілених web застосунків.*	*Object-relational mapping (ORM) у веб-застосунку на основі патерну MVT фреймворку (Python Django). Управління базами даних у фреймворку Django (mysite/settings.py). Створення та активація моделі бази даних у фреймворку Django. Управління базою даних через API Django. Створення, налаштування та тестування адміністративної панелі. Тестування моделі бази даних ORM через адміністративну панель. Технології JavaSripth для роботи з базами даних. Використання контейнера Docker для створення програмної інфраструктури, що відокремлює базу даних та вебзастосунок.* *Завдання на СРС: п.6 №7*
8	*Лекція 6. Програмна архітектура Entity Framework для роботи з базою даних типових розподілених web застосунків.*	*Альтернативні технології для роботи з базами даних у вебзастосунку на основі патерну MVT фреймворку (Python Django). Класифікація технології для роботи з базами даних у .NET Core. Object-relational mapping (ORM) у вебзастосунку на основі патерну MVC фреймворку (.NET Core). Entity Framework у технології .NET. Технології JavaSripth для роботи з базами даних* *Завдання на СРС: п.6 №8, 32.*
9	*Практикум № 2.*	*Розгортання вебзастосунку у мережі на безкоштовному хостингу.* *Завдання на СРС: п.6 №9.*
Тема 3. Система контейнеризації як основа створення складної розподіленої інфраструктури програмного забезпечення web-застосунку.		

10	*Лекція 7. Загальне налаштування прототипу типового розподіленого web-застосунку для роботи у контейнері.*	*Установка базових програмних складових інфраструктури для роботи з системою контейнерів у Django. Система відображення на основі патерну MVT фреймворку (Python Django). Система управління запитами на основі MVT фреймворку (mysite.urls). Установка базових програмних складових інфраструктури для роботи з системою контейнерів у .NET Core. Node.js, NPM Package, Git, Docker,.NET Core Software Development Kit. Система відображення на основі патерну MVC фреймворку (.NET Core). Система управління запитами на основі MVC фреймворку (.NET Core). Завдання на СРС: п.6 №10.*
11	*Лекція 8. Тестування прототипу типового розподіленого web-застосунку для роботи у контейнері.*	*Тестування базових програмних складових інфраструктури для роботи з системою контейнерів у Django.* *Завдання на СРС:. п.6 №11.*
12	*Практикум № 3.1*	*Проектування контейнера Docker для розгортання та розробки вебзастосунку. Завдання на СРС:. п.6 №12.*
Тема 4. Автоматизація управління програмною інфраструктурою у системі контейнеризації		
13	*Лекція 9. Основи розподіленої програмної інфраструктури на базі системи контейнеризації.*	*Загальна архітектура системи контейнерізації на базі Docker. Управління образами докера. Стани контейнерів. Базові команди nf приклади використання докера на основі типового розподіленого web-застосунку.* *Завдання на СРС: п.6 №13.*
14	*Лекція 10. Автоматизація управління з використанням Dockerfile.*	*Основи технології роботи з Dockerfile. Синтаксис та правила створення Dockerfile. Практика використання Dockerfile для складної програмної інфраструктури.* *Завдання на СРС:. п.6 №14.*
15	*Практикум № 3.2*	*Проектування контейнера Docker для розгортання та розробки вебзастосунку. Завдання на СРС:. п.6 №15.*
16	*Лекція 11. Базові елементи мережі Docker.*	*Загальна програмна інфраструктура розподілених вебзастосунків у мережі Docker. Управління мережею Docker.* *Завдання на СРС:. п.6 №16.*
17	*Лекція 12. Автоматизація управління складною програмною інфраструктурою на базі технології Docker Compose і Swarm mode.*	*Інсталяція системи Docker Compose. Запуск та зупинка системи Docker Compose. Базовий синтаксис Docker compose file. Автоматизація управління з використанням Swarm mode. Масштабування складної програмної інфраструктури. Тестування складної програмної інфраструктури. Управління сервісами у складній програмній інфраструктурі. Балансування*

		навантаження у складній програмній інфраструктурі. Команди Manage Services. *Завдання на СРС: п.6 №17.*
18	*Практикум № 4.1*	*Декомпозиція складної програмної архітектури вебзастосунку у вигляді окремих сервісів і розробка мікросервісної архітектури.* *Завдання на СРС: п.6 №18.*
Тема 5. Мікросервісна архітектура як основа створення програмної інфраструктури розподілених веб-застосунків		
19	*Лекція 13. Основи мікросервісної архітектури.*	*Загальний опис мікросервісної архітектури. Переваги мікросервісної архітектури. Недоліки мікросервісної архітектури.* *Завдання на СРС: п.6 №19.*
20	*Лекція 14. Програмна інфраструктура на основі мікросервісів.*	*Горизонтальне масштабування програмної інфраструктури на основі мікросервісів (X-Axis Scaling). Вертикальне масштабування програмної інфраструктури на основі мікросервісів (Y-Axis Scaling). Масштабування програмної інфраструктури на основі мікросервісів (Z-Axis Scaling).* *Завдання на СРС: п.6 №20.*
21	*Лекція 15. Типові шаблони мікросервісної архітектури у складній розподіленій інфраструктури.*	*Шаблон (патерн) агрегатор Aggregator Pattern.* *Шаблон (патерн) проксі Proxy Pattern.* *Шаблон (патерн) Chained Pattern.* *Шаблон (патерн) Branch Microservice Pattern.* *Шаблон (патерн) Shared Resource Pattern.* *Завдання на СРС: п.6 №21.*
22	*Практикум № 4.2*	*Декомпозиція складної програмної архітектури веб-застосунку у вигляді окремих сервісів і розробка мікросервісної архітектури.* *Завдання на СРС: п.6 №22.*
23	*Лекція 16. Лекція-семінар. Доповіді за тематикою розподіленої інфраструктури.*	*Завдання на СРС: п.6 №23.*
24	*Практикум № 5.*	*Створення програмної інфраструктури на основі контейнерів Docker для розгортання і управління складною програмною архітектурою .* *Завдання на СРС: п.6 №24.*
25	*Лекція 17. Лекція-семінар. Доповіді за тематикою управління розподіленою інфраструктурою.*	*Завдання на СРС: п.6 №25.*
Модульна контрольна робота		

6. Самостійна робота студента/аспіранта

Дисципліна ґрунтується на самостійних підготовках до аудиторних занять на теоретичні та практичні теми.

№ з/п	*Назва теми, що виноситься на самостійне опрацювання*	*Кількість годин*	*Література*
1	*Підготовка до лекції 1*	*1*	*1;1-26*
2	*Підготовка до лекції 2*	*1*	*1;1-26*
3	*Підготовка до комп'ютерного практикуму 1.1*	*1,5*	*1;1-26*
4	*Підготовка до лекції 3*	*1*	*1;1-26*
5	*Підготовка до лекції 4*	*1*	*1;1-26*
6	*Підготовка до комп'ютерного практикуму 1.2*	*1,5*	*1;1-26*
7	*Підготовка до лекції 5*	*1*	*1;1-26*
8	*Підготовка до лекції 6*	*1*	*1;1-26*
9	*Підготовка до комп'ютерного практикуму 2*	*1,5*	*1;1-26*
10	*Підготовка до лекції 7*	*1*	*1;1-26*
11	*Підготовка до лекції 8*	*1*	*1;1-26*
12	*Підготовка до комп'ютерного практикуму 3.1*	*1,5*	*1;1-26*
13	*Підготовка до лекції 9*	*1*	*1;1-26*
14	*Підготовка до лекції 10*	*1*	*1;1-26*
15	*Підготовка до комп'ютерного практикуму 3.2*	*1,5*	*1;1-26*
16	*Підготовка до лекції 11*	*1*	*1;1-26*
17	*Підготовка до лекції 12*	*1*	*1;1-26*
18	*Підготовка до комп'ютерного практикуму 4.1*	*1,5*	*1;1-26*
19	*Підготовка до лекції 13*	*1*	*1;1-26*
20	*Підготовка до лекції 14*	*1*	*1;1-26*
21	*Підготовка до лекції 15*	*1*	*1;1-26*
22	*Підготовка до комп'ютерного практикуму 4.2*	*1,5*	*1;1-26*
23	*Підготовка до лекції 16 (лекція-семінар)*	*3*	*1, 1-26*
24	*Підготовка до комп'ютерного практикуму 5*	*1,5*	*1, 1-26*
25	*Підготовка до лекції 17 (лекція-семінар)*	*3*	*1, 1-26*
26	*Підготовка до модульної контрольної роботи*	*6*	*1, 1-26*
27	*Підготовка до заліку*	*6*	*1, 1-26*
28	*Перелік фреймворків для створення вебзастосунків на базі різних мов програмування.*	*5*	*1*
29	*Розподілені системи контролю версій.*	*4*	*1*
30	*Технології і практика роботи з HTML5 та супутні*	*5*	*27*

	технології.		
31	*Основи організації комп'ютерних мереж.*	*4*	*27*

Політика та контроль

7. Політика навчальної дисципліни (освітнього компонента)

Відвідування лекційних занять є обов'язковим. В умовах особливого стану питання щодо відвідування лекційних занять та інших аспектів політики може бути змінені.

Відвідування занять комп'ютерного практикуму може бути епізодичним та за потреби консультації/захисту робіт комп'ютерного практикуму.

Правила поведінки на заняттях: активність, повага до присутніх, відключення телефонів.

Дотримання політики академічної доброчесності.

Правила захисту робіт комп'ютерного практикуму: роботи повинні бути зроблені відповідно до поставлених задач та згідно з варіантом.

Правила призначення заохочувальних та штрафних балів є наступними. Заохочувальні бали нараховуються за:
- точні та повні відповіді в опитуваннях за матеріалами лекцій (максимальна кількість балів за опитування - 3 бали).

8. Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Протягом семестру студенти виконують 5 комп'ютерних практикумів. Максимальна кількість балів за кожний комп'ютерний практикум: 10 балів.

Бали нараховуються за:
- якість виконання комп'ютерного практикуму: 0-5 бали;
- відповідь під час захисту комп'ютерного практикуму: 0-3 бали;
- своєчасне представлення роботи до захисту: 0-2 бали.

Критерії оцінювання якості виконання:
5 бали – робота виконана якісно, в повному обсязі;
4 бали – робота виконана якісно, в повному обсязі, але має недоліки;
3 бали – робота виконана в повному обсязі, але містить незначні помилки;
2 бали – робота виконана в повному обсязі, але містить суттєві помилки;
0 балів – робота виконана не в повному обсязі.

Критерії оцінювання відповіді:
3 бали – відповідь повна, добре аргументована;
2 бали –відповідь вірна, але має недоліки або незначні помилки;
1 бал – у відповіді є суттєві помилки;
0 балів – немає відповіді або відповідь невірна.

Критерії оцінювання своєчасності представлення роботи до захисту:
2 бали – робота представлена до захисту не пізніше вказаного терміну;
0 балів – робота представлена до захисту пізніше вказаного терміну.

Максимальна кількість балів за виконання та захист комп'ютерних практикумів:
10 балів × 5 комп. практ. = 50 балів.

*Протягом семестру на лекціях відбуваються **опитування за темою поточного заняття**. Максимальна кількість балів за всі опитування: 3 бали. Кількість **опитування за темою поточного заняття** для одного студента є необмеженою.*

*Завдання на **модульну контрольну роботу** складається з 3 теоретичних та 2 практичних запитань. Відповідь на кожне запитання оцінюється 10 балами.*

Критерії оцінювання кожного запитання контрольної роботи:
9-10 балів – відповідь вірна, повна, добре аргументована;
7-8 балів – відповідь вірна, розгорнута, але не дуже добре аргументована;
5-6 балів – в цілому відповідь вірна, але має недоліки;
3-4 балів – у відповіді є незначні помилки;
1-2 бали – у відповіді є суттєві помилки;
0 балів – немає відповіді або відповідь невірна.

Максимальна кількість балів за модульну контрольну роботу:
10 балів × 5 запитань = 50 балів.

Рейтингова шкала з дисципліни дорівнює:
$R = R_C$ = 50 балів + 50 балів = 100 балів.

Календарний контроль: проводиться двічі на семестр як моніторинг поточного стану виконання вимог силабусу.
На першій атестації (8-й тиждень) студент отримує «зараховано», якщо його поточний рейтинг не менше 10 балів (50 % від максимальної кількості балів, яку може отримати студент до першої атестації).
На другій атестації (14-й тиждень) студент отримує «зараховано», якщо його поточний рейтинг не менше 15 балів (50 % від максимальної кількості балів, яку може отримати студент до другої атестації).

Семестровий контроль: залік
Умови допуску до семестрового контролю:
При семестровому рейтингу (R_C) не менше 60 балів та зарахуванні усіх робіт комп'ютерного практикуму, студент отримує залік «автоматом» відповідно до таблиці (Таблиця відповідності рейтингових балів оцінкам за університетською шкалою). В іншому разі він має виконувати залікову контрольну роботу.

Необхідною умовою допуску до залікової контрольної роботи є виконання і захист комп'ютерного практикуму.

Якщо студент не погоджується з оцінкою «автоматом», то може спробувати підвищити свою оцінку шляхом написання залікової контрольної роботи, при цьому його бали, отримані за семестр, зберігаються, а з двох отриманих студентом оцінок виставляється краща («м'яка» система оцінювання).

Таблиця відповідності рейтингових балів оцінкам за університетською шкалою:

Кількість балів	*Оцінка*
100-95	Відмінно
94-85	Дуже добре
84-75	Добре
74-65	Задовільно
64-60	Достатньо
Менше 60	Незадовільно
Не виконані умови допуску	Не допущено

9. Додаткова інформація з дисципліни (освітнього компонента)

Для роботи можуть бути використані різні операційні системи за бажанням студента і по узгодженню з викладачем.
У випадку бажання студента використовувати у комп'ютерному практикумі інші мови програмування або технології вони можуть бути включені як додатковий елемент у базову архітектуру розподіленого вебзастосунку.

Робочу програму навчальної дисципліни (силабус):

Складено к.т.н., доцент, Катін П.Ю.

Ухвалено кафедрою ПЗКС (протокол №8 від 25.01.2023)

Погоджено Методичною комісією факультету прикладної математики (протокол №6 від 27.01.2023)